NOD32 for Linux Mail Servers

Installation Manual
and
Users' Documentation

Copyright 2005, Eset, s.r.o.

NOD32 for Linux Mail Server , First Edition
Published on 11th March 2005
Copyright © 2005 Eset s.r.o.

All rights reserved. No part of this documentation may be reproduced, stored in a retrieval system or transmitted in any form or by any means, electronic, mechanical, photocopying, recording, scanning, or otherwise without a permission in writing from the author. Eset s.r.o. reserves the right to change any of the described application software without prior notice.

Revision History

Revision 2.14-1 (11/03/2005)
Chapter with alternative methods using NOD32 Command Line Interface (nod32cli) added.
Revision 2.13-2 (10/02/2005)
Chapter with tips and tricks for NOD32LMS configuration added.
Revision 2.13-1 (01/01/2005)
Authorized transcription of NOD32 for Linux Mail Servers (version 2.13-1) user's guide.

Table of Contents

1. Introduction **1**
2. How to navigate through this guide **3**
3. Mail server in UNIX OS environment **5**
4. NOD32LMS package installation **9**
5. NOD32LMS basic configuration **13**
5.1. NOD32LMS - own configuration 13
5.2. Scanning of the inbound e-mail messages 15
5.2.1. Renaming the original MDA and its replacement by NOD32MDA 16
5.2.2. Setting of NOD32MDA (in MTA) as MDA 18
5.2.2.1. Setting Sendmail MTA 18
5.2.2.2. Setting Postfix MTA 19
5.2.2.3. Setting Qmail MTA 20
5.2.2.4. Setting MTA Exim version 3 22
5.2.2.5. Setting MTA Exim version 4 23
5.3. Scanning the outbound e-mail messages 25
5.4. Content filtering in MTA 28
5.4.1. Content filtering in MTA Postfix 28
5.4.2. Content filtering in MTA Sendmail 29
5.5. Alternative methods of scanning e-mails. 30
5.5.1. Scanning e-mail messages using AMaViS 30
5.5.1.1. amavis 31
5.5.1.2. amavisd 32
5.5.1.3. amavisd-new 32
6. NOD32 system update and maintenance **33**
6.1. Basic concept of NOD32 system update. 33
6.1.1. NOD32 modules organization 33
6.1.2. NOD32 mirror creation 34
6.1.3. Generation of NOD32 scanner loading modules 34
6.2. Subordinate mirrors creation 35
6.3. Automatic update of the virus definitions database 36
6.3.1. Structure and use of automatic update script 36
6.3.2. Periodic update of the virus definitions database 37
7. Tips and tricks **39**
7.1. Dropping messages marked by NOD32 as deleted in MTA Postfix 39
7.2. NOD32LMS and TLS support in MTA 39
8. Let us know **43**
A. Installed content of NOD32LMS package **45**

Chapter 1. Introduction

Dear user, you have acquired NOD32 for Linux Mail Servers - NOD32LMS - probably the best antivirus system for e-mail servers running under the Linux operating system. As you will soon realize, the system has unsurpassed scanning speed and detection rate, combined with a very small footprint that makes NOD32 the ideal choice for any Linux mail server.

The following is a short summary of its key features.

- NOD32 scanning engine algorithms provide both the highest detection rate and the fastest scanning times.
- It includes unique advanced heuristics for Win32 worms and backdoors.
- The system solution is MTA-independent (mail server independent).
- Inbuilt NOD32 archivers unpack archived e-mail attachments without the need for any external programs.
- In order to increase speed and efficiency of the system, its architecture is based on the running daemon (resident program) where all the scanning requests are sent to.
- Six various levels of logging can be configured to get information about system activity and infiltrations.
- One of the major advantages is the fact that the system installation does not require external libraries or programs except for libc.
- The system can be configured to notify any person in case of detected infiltration.
- Moreover, information about infiltration can be configured to be written into an e-mail header, footer and subject.

To run efficiently, the NOD32LMS requires just 5MB of hard-disk space and 8MB of RAM. The system runs smoothly under the 2.2.x, 2.4.x and 2.6.x Linux OS kernel versions. It supports most popular mail server software including Sendmail, Postfix, Qmail, Exim, etc. and runs under all of the common Linux distributions:

- Fedora Core 1 and higher,
- Red Hat version 5.x and higher,
- Mandrake version 5.x and higher,
- SuSE version 6.x and higher,
- Debian version 2.x and higher.

From lower-powered, small office mail servers to enterprise-class ISP mail servers with thousands of users, NOD32LMS delivers the performance and scalability you expect from a UNIX based solution and the unequalled security of NOD32.

Chapter 2. How to navigate through this guide

This guide is assumed to be the complex users' guide into the NOD32LMS system. It covers information on configuration and maintenance of the system in order to run efficiently for various supported Linux OS distributions and various e-mail server systems.

Therefore, assuming you are a typical system administrator being too busy to read all these pages, here are some hints on how to surf through this guide with just catching the most relevant information necessary to setup and run.

First, look into the common messaging system scheme (next section) to see what the NOD32LMS is able to do for you. Identify the proper implementation of the NOD32LMS into your system and skip directly to the section that discusses the appropriate scenario.

Second, identify an e-mail server system running on your Linux OS and browse through just the appropriate sections of this guide. All the guide sections are done in a manner that the e-mail server system name always appears in the section title, in case the section is system specific.

Third, look into the foreword to each chapter to get information related to relevance of the information stored in the chapter. Often you will get a hint to skip directly to the section that contains just the step by step statement information in order to suffice what is necessary.

Fourth, besides the information in this guide, there exists manual pages related to the individual components of the NOD32LMS package. The pages are always a useful reference for getting actual information on the accurate components setting options that are not all mentioned here.

Lastly, note that the guide is divided in a following manner:

Chapter 1

Introduction.

Chapter 2

This text, that you have almost finished reading.

Chapter 3

Common scheme of the UNIX OS messaging system. Gives you an overview on what NOD32LMS can do for you.

Chapter 4

Leads you step by step to install the system.

Chapter 5

Lists e-mail server specific configurations.

Chapter 6

Will help you keep your system up to date.

Chapter 7

Contains tips and tricks on configuration of NOD32LMS.

Chapter 8

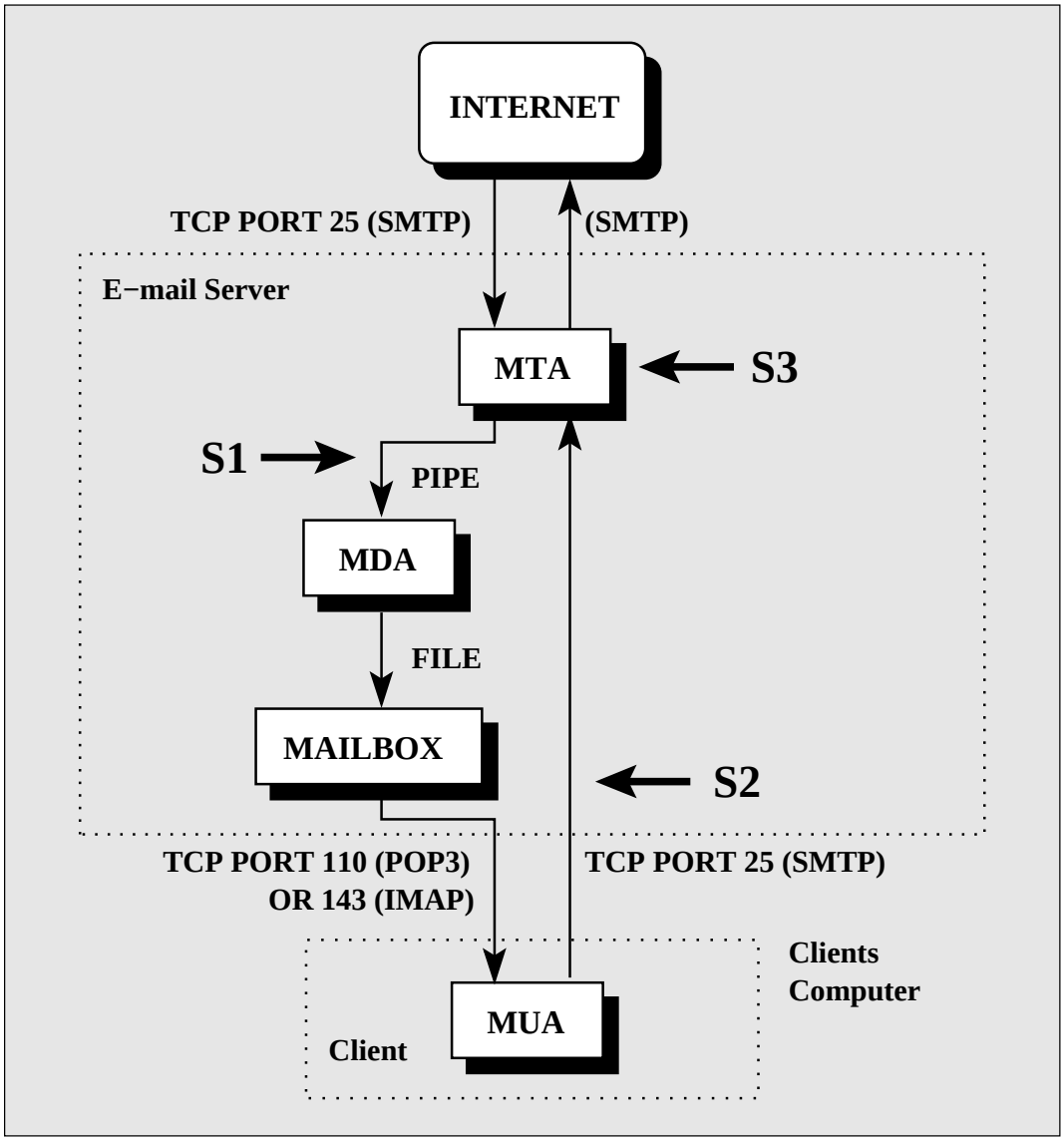
Contains information on where to send your questions or remarks.

Chapter 3. Mail server in UNIX OS environment

This chapter is concerned with the basics of the e-mail messaging system, also commonly called e-mail server system, however, e-mail server is only part of the more complex messaging system. For better understanding of the NOD32LMS operation, knowledge of the messaging system basic principles is of paramount importance. Therefore we do not recommend to skipping this section unless this knowledge is already acquired.

The following diagram is a rough scheme of the UNIX OS e-mail messaging system.

Figure 3-1. Scheme of UNIX OS e-mail messaging system.



The meaning of abbreviations used in the scheme of figure 3-1 is as follows.

MTA (Mail Transport Agent)

A program (for instance sendmail, postfix, qmail, exim, etc.) receives e-mail messages from local and/or remote domains and forwards it for further delivery. Generally speaking, MTA is an agent providing mail transfer among other e-mail servers MTAs and/or MUAs (see below).

MDA (Mail Delivery Agent)

A program (maildrop, procmail, deliver, local.mail, etc.) providing delivery of an e-mail into a particular mailbox.

MUA (Mail User Agent)

An e-mail processing program (MS Outlook, Mozilla Mail, Eudora, etc.) that allows user to access and manage e-mail messages (i.e. read, compose, print them etc.).

MAILBOX

A file or a file structure on a disk serving as the storage space for e-mails. Note: There are several formats of Mailboxes in Linux OS. (e.g.: an old fashioned format where e-mails for each user are stored sequentially in one user appropriate file located in directory /var/spool/mail; MBOX (a bit newer but still an old format) with e-mails stored sequentially in one file located within user home directory; MAILDIR with e-mail stored in a separate file within a hierarchical directory structure.

Now the scheme in the figure 3-1 represents a typical e-mail gateway placed at an entrance to some local network. This means that the e-mail server receives data communication typically via TCP port 25 (SMTP - Simple Mail Transfer Protocol is used within this process). The message received is transfered by the local MTA either to another remote e-mail server system or the message is delivered by using local MDA into the appropriate MAILBOX (we assume that each user belonging to the local network has a corresponding MAILBOX located at the server). It is then a responsibility of the client's local MUA to provide download and/or correct interpretation of the message at the client's computer. To get the data from an e-mail server system the MUA uses typically TCP port 110 (POP3 - Post Office Protocol) or TCP port 143 (IMAP - Internet Message Access Protocol). On the other hand if a user at the client's computer would like to send an e-mail message to the Internet, it is again the responsibility of the local MUA to deliver the message via TCP port 25 (SMTP) to the local MTA (located at an entrance to the local network) that will take care of the further message delivery.

The operating principle of NOD32LMS system is based on the idea of data communication interception at the various phases of its transfer and of scanning this communication by NOD32 scanning engine. Those locations are marked in the figure 3-1 by symbols S1, S2 and S3. In the following text we will distinguish between three scenarios of e-mail message scanning which basically corresponds to the referred marks:

- Scanning of inbound e-mail messages (We define the term "inbound message" for e-mail message with the target address corresponding to the destination located at the local domain. Similarly the "outbound message" will be a message bound to some remote domain via its target address.) marked in the figure by symbol S1, is used to protect e-mail messages delivered from the outside Internet to the local MAILBOX-es belonging to local users.
- Scanning of outbound messages marked in the figure by symbol S2 is used to protect the e-mail messages sent by local user MUAs to the outside Internet.
- Bi-directional scanning (usually known also as content filtering in MTA) marked by symbol S3 is devoted to check both directions (or even all directions) of the e-mail messages flow.

Note: Bi-directional scanning scenario is more complex than the other two methods previously discussed. The bi-directional scanning checks not only message flow directed from Internet to a local network or vice versa. Indeed, this scanning also takes care of the messages that will come to the gateway server and are further routed to the different remote domains. Therefore in a general case this kind of scanning can be used typically as a checkpoint between two different Internet networks. From this point of view, concerning our local e-mail gateway, this kind of scanning is not necessarily required, as this creates a much higher load on the server computer than previously discussed scenarios.

Chapter 4. NOD32LMS package installation

Before further explanations concerned with NOD32LMS, let's first install the whole thing. In order to do so, one has to download the appropriate packages from the NOD32 server. Use your favorite web browser to navigate to the NOD32 download page

```
http://www.nod32.com/download/download.htm
```

At this page you can see a set of NOD32LMS packages listed for various UNIX OS distributions. Identify the appropriate distribution of the actual OS installed on your computer and download the appropriate package.

Note: For the download process to succeed, the username and password for authentication against the NOD32 server is required. You will receive both from your vendor.

After successful download the whole installation relies on one command line statement written in the directory where the package is downloaded. In case the OS distribution installed at your computer is one of the following (Fedora, Red Hat, Suse, Mandrake, or other Linux OS using Red Hat package manager) you need to write the statement:

```
rpm -i nod32lms-x.xx-x.i386.rpm
```

In case the NOD32LMS is already installed on your computer you will receive the following or similar message in return.

```
package nod32lms-x.xx-x.i386.rpm is already installed
```

In this case you have to use a different switch -u to update the appropriate rpm package, so the statement will look like this

```
rpm -u nod32lms-x.xx-x.i386.rpm
```

The Debian Linux OS uses its own package manager for installation, so called dpkg (debian packager) and the files are also of different format (suffix deb). So the appropriate statement for installation and/or update of the package in this case will be as follows

```
dpkg -i nod32lms-x.xx-x.i386.deb
```

Once the NOD32LMS package is installed, one can see that the main NOD32LMS scanning daemon nod32d is running in the background. Indeed, the installation apparatus is done in a way that it starts the daemon automatically. One can also see that from now on the nod32d daemon will also start immediately after a computer boot. To be absolutely certain, the fact that the daemon is running can be checked by typing the command

```
ps -C nod32d
```

that should return the following or simillar message

```
PID TTY          TIME CMD
2226 ?        00:00:00 nod32d
```

If this is not the case, then something is wrong and you should report this as a bug to the NOD32 support service.

As you will soon realize, there are also other daemons within a NOD32LMS package that are required to run in some system configurations. For instance the daemon nod32smtp or nod32smfi. However, since the functionality of these daemons is not required in all cases, the installation script does not provide the automatic start-up of them at system boot. If necessary one has to do it by hand. Here is the description of how to do this.

In order to start, for instance, nod32smtp daemon under Red Hat or Mandrake, one has to write the following command

```
/sbin/service nod32smtp start
```

Under SuSE the appropriate command would be

```
/sbin/startproc /usr/bin/nod32smtp
```

Finally, for the Debian the equivalent command is

```
/sbin/start-stop-daemon --start --exec /usr/sbin/nod32smtp
```

Similarly, for stopping the appropriate service, one would write under RedHat or Mandrake the following command

```
/sbin/service nod32smtp stop
```

Under SuSE the appropriate command would be

```
/sbin/killproc /usr/bin/nod32smtp
```

and for Debian the equivalent command is

```
/sbin/start-stop-daemon --stop --name nod32smtp
```

Above are the useful commands always used when reconfiguring the system (in this case the NOD32LMS system always has to be restarted).

Now we will describe how to provide the automatic start of the daemon at system boot. In order to do so, one has to write under Red Hat or Mandrake system the following commands

```
/sbin/chkconfig --add nod32smtp
```

Under SuSE Linux the following command will be valid

```
/sbin/insserv /etc/init.d/nod32smtp
```

and finally for the Debian the statement

```
/usr/sbin/update-rc.d nod32smtp defaults
```

can be used. After using these commands, the system will create all the necessary settings for the automatic start of the nod32smtp daemon at system boot.

However, we still have to discuss one circumstance about the NOD32LMS package installation. In case none of the above OS distributions correspond to your actual operating system, you can try to install NOD32LMS from the tgz (tarred and gzipped) format. The rest of this chapter is a step by step guide to do so, therefore if this is not relevant to you, skip directly to the next section.

In order to install NOD32LMS from a generic tgz file, copy the file nod32lms-x.xx-x.i386.lnx.tgz, for instance, into the directory /usr/local/src. Unpack the file

```
tar -xvzf nod32lms-x.xx-x.i386.lnx.tgz
```

and copy the individual components of the package into the corresponding subdirectories of the root directory, i.e. change current working directory to the unpacked package directory

```
cd /usr/local/src/nod32lms-x.xx
```

and write the following statement

```
cp -r * /
```

The switch -r will provide the recursive copying of all package components to the proper place in the root directory. If this is not the case or implementation of copy (cp) statement does not provide this functionality, you should probably copy individual components one by one. For this purpose check the appendix A to see complete NOD32LMS package content installed at the proper location.

Note: After copying the NOD32LMS components you have to adjust the initialization scripts for the daemon programs of NOD32LMS. In the /etc/init.d directory you should see a short scripts, corresponding to the individual NOD32LMS components and OS distributions by their names. For instance

```
/etc/init.d/nod32d.deb
```

is the initialization script for nod32d daemon used for Debian Linux distribution. Try adjusting one of the scripts for each daemon in order to make it able to start.

Chapter 5. NOD32LMS basic configuration

This chapter describes the process of the NOD32LMS configuration. In the first section the structure and meaning of all NOD32LMS configuration files will be discussed. The rest of the chapter is devoted to individual scenarios related configurations as discussed briefly in chapter 3.

5.1. NOD32LMS - own configuration

All the necessary configuration files concerned with functionality of NOD32LMS are stored within a directory

```
/etc/nod32
```

In the following text all the files stored within this directory are going to be discussed.

/etc/nod32/nod32.cfg

This is a most important file, commonly known also as 'main NOD32 configuration file'. After exploring the file you can see that its structure is composed from various parameters of NOD32LMS that are all distributed within sections (note the section names are always enclosed in square brackets - []) . There is always one global and several special sections in the file. The configuration of the whole system (i.e. key features discussed within Introduction to this guide and much more) is provided by setting parameters used within individual sections. The parameters defined within a global section are valid for all modules (sometimes we call those modules 'agents') of NOD32LMS. The parameters defined within a special section, on the other hand, are valid only for an appropriate agent of NOD32LMS. At this point we should describe the individual parameters meaning, however, we will not do it here as it is done in manual pages corresponding to the appropriate module. For instance the parameters that can be used in global and also in special sections are described in the manual page concerned with daemon nod32d (main NOD32LMS daemon). The manual page can be invoked using the following statement

```
man nod32d
```

Similarly, the parameters that can be only used in the special sections are described in the manual pages concerned with individual modules, i.e. nod32mda, nod32smtp, nod32smfi and nod32cli.

We recommend the user to use these resources always when referring to the parameters of NOD32LMS components. Refer to the appendix A to see a full list of manual pages installed.

Let's return to the description of the structure of the main configuration file. It is good to note that every parameter listed in the global section has a lower priority than the same parameter listed in the module section. The only exception is concerned with the parameters reported in the manual page for daemon nod32d as private, which are used only by nod32d daemon. These are typically the logging parameters as there is only daemon nod32d that provides you with logging of NOD32LMS activity and infiltrations detected by the system.

To finish a general description of the main configuration file, we will yet make a remark on the parameters type. Note that there are basically three types of parameters in the configuration file, e.g.:

integer

Integer parameters accept integer values. For instance (listen_port = 2526).

string

String parameters accept the strings delimited by quotation marks (server_addr = "localhost").

logical variable

Logical variables accept two values: yes or no. For instance (scan_obj_emails = yes) or (scan_obj_emails = no).

/etc/nod32/nod32.auth

This is the so called 'user authentication file' that is used to authenticate the user against the NOD32 server when making an update of the virus signatures database. We will not discuss the meaning of the file here as it is done later in chapter 6 with more details.

/etc/nod32/sig_*.html.example

When browsing through the /etc/nod32 directory you have probably noticed a set of files whose names are starting with same prefix 'sig' and ending with suffix 'html.example'. These are the html templates that can be used to define the text, inserted as a footnote, into the scanned e-mail messages (use your favorite editor to explore the files). In order to enable this feature, you have to set an appropriate option in the main configuration file (write_to_emails) that can be defined in a global section of the file as well as in the individual modules section of the file depending on what level of footnote writing we would like to achieve in appropriate individual modules (see nod32d manual page for detail). The other requirement to enable the custom html templates is to remove the suffix '.example' from all the file names of the templates. In this case the NOD32LMS system will ignore the default footnote text and will use the text predefined in the html files instead. The meaning of the individual files can be explained directly in the structure of e-mail footnotes inserted.

When NOD32 system has detected infiltration in the e-mail message, the following footnote template is written into the footnote when enabled.

```
e-mail header   | From:
.               | To:
----------------------------
e-mail body     |
.               |
.               |html text of the file sig_header_infected.html
.               |
.               |list of infiltrations found by the scanner
.               |
.               |html text of the file sig_footer_infected.html
```

When NOD32 system has not detected any infiltration in the e-mail message, the following footnote template is written into the footnote when enabled.

```
e-mail header   | From:
.               | To:
----------------------------
e-mail body     |
.               |
.               |html text of the file sig_header_clean.html
.               |
.               |list of infiltrations found by the scanner
.               |
.               |html text of the file sig_footer_clean.html
```

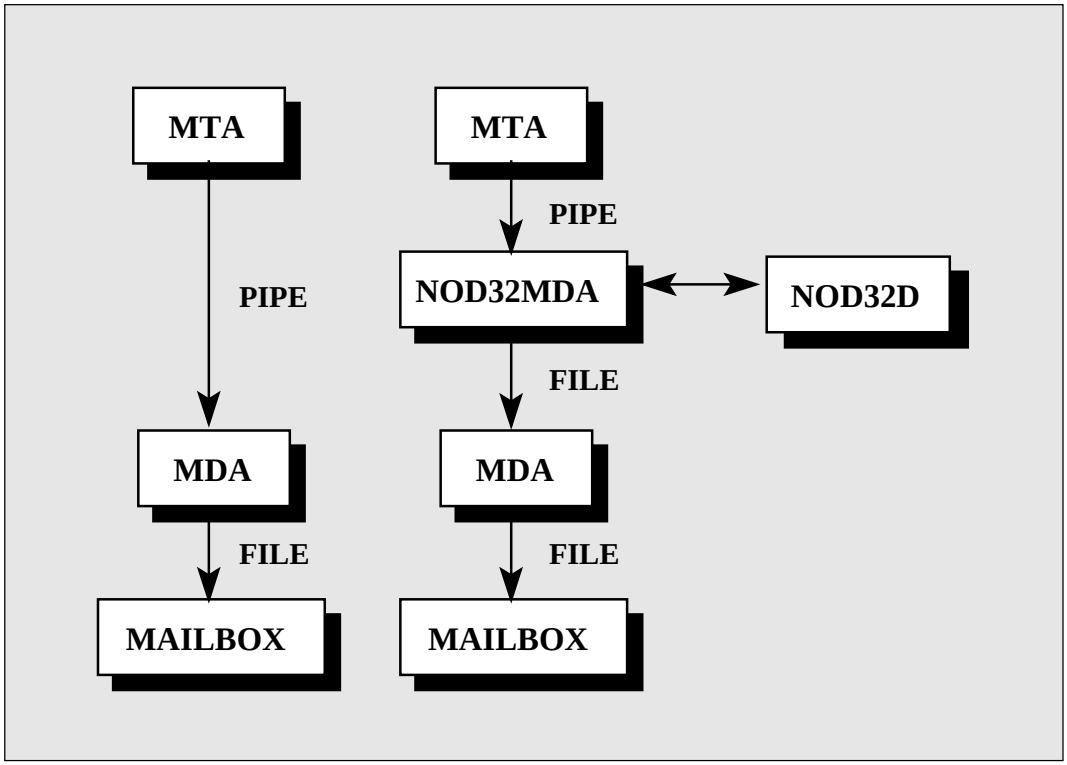
/etc/nod32/nod32d_script

There is one more file within /etc/nod32 directory. It is a NOD32LMS notification script. In case NOD32 scanner has found an infiltration within a scanned message, this script (if enabled) is used to send a notification about the event to any person defined within a script. This functionality is enabled by logical parameter of the main configuration file (exec_script = yes). To get more information on this topic, read the section NOTIFICATION within the nod32d manual pages.

5.2. Scanning of the inbound e-mail messages

Scanning of the inbound e-mail messages is performed at the time of message transmission between MTA and MDA (as marked by symbol S1 in figure 3-1). The generic scheme of the process is shown in the figure 5-1.

Figure 5-1. The scheme of the inbound e-mail message delivery without (left part of the figure) and with (right part of the figure) the NOD32 scanning.



The incoming e-mail is scanned by nod32mda module supported by daemon nod32d. Scanned e-mail is afterwards forwarded to the original MDA for further delivery to the MAILBOX.

As shown in the figure 5-1, the virus scanning feature can be enabled by appropriate configuration setting of the MTA agent and nod32mda module. It is also apparent from the figure that the solution is MDA independent.

Note: The majority of mail servers use procmail or maildrop (MDA). The nod32mda module supports any MDA. The following MDAs were tested: procmail, maildrop, deliver and local.mail.

In general, the virus scanning feature can be enabled in two ways described below.

5.2.1. Renaming the original MDA and its replacement by NOD32MDA

This is a simple approach even without a need to make any changes in the agent MTA. But still, prior starting with the proper setup modification, the user is required to know exactly what MDA agent is used by the system. This information can be grabbed only by exploring the MTA configuration file. More information on how to do this is described shortly for individual MTAs always at the start of the appropriate subsections 5.2.2.1. (sendmail), 5.2.2.2. (postfix), 5.2.2.3. (qmail), 5.2.2.4. (exim version 3 or less) and 5.2.2.4. (exim version 4). Please, check the appropriate text parts and then return here. For illustration purposes, we will use the MDA procmail.

In the next step we have to find the location of the MDA. In order to do so we use the following command:

```
which procmail
```

that will give as the full path to agent MDA in return. Now the most important part of the modification is to rename the original procmail binary file to procmail.real:

```
mv /usr/bin/procmail /usr/bin/procmail.real
```

and create the soft link to module nod32mda with the name 'procmail' located in the directory where the original MDA procmail was found, i.e. with the name '/usr/bin/procmail':

```
ln -s /usr/bin/nod32mda /usr/bin/procmail
```

With the above modifications, we have ensured that all the messages originally sent by the MTA for delivery to MAILBOX (by using an original MDA) will now be primarily sent to module nod32mda. Still there remains the second part of the modification to provide that all messages processed by nod32mda will be sent to the original procmail MDA (at this time renamed to procmail.real). In order to do so, we have to modify parameter 'mda_path' within the section [mda] of the main NOD32LMS configuration file (/etc/nod32/nod32.cfg) as follows:

```
mda_path = "/usr/bin/procmail.real"
```

After all of the above modifications one has to restart the daemon nod32d in order to reread the new configuration of NOD32LMS.

Note that in the above example we have used procmail as an MDA, however, the whole procedure can be done with any other known MDA which is a great advantage of this approach. On the other hand, there is a potential problem within this configuration that may occur in circumstances of MDA software upgrade. Indeed, once we will decide to upgrade the MDA software (for instance procmail in our case), the upgrading mechanism will provide the new binary file /usr/bin/procmail that will cancel the link to nod32mda scanning module created by the above procedure and the configuration will be broken. Thus, even if the approach suggested in this section works in all cases, it is always necessary keep in mind that the problem of the

link cancellation may occur and one has always, after the MDA software upgrade, to repeat the whole procedure described above.

5.2.2. Setting of NOD32MDA (in MTA) as MDA

This section contains a more rigorous approach to provide scanning of inbound messages than the one described in the previous section. Even if the approach described here requires modification of the MTA configuration, we recommend it due to the fact that it does not bring the potential problems, described at the end of the previous section, to the system.

Note: Some MTA modules may be configured to not need an MDA component to deliver e-mails. In such (very rare) cases, it is necessary to configure the MTA to use any, by NOD32LMS supported, external MDA (for instance procmail or maildrop).

5.2.2.1. Setting Sendmail MTA

An MTA Sendmail stores all its configurations in one file /etc/mail/sendmail.cf (in older operating systems it can be stored in /etc/sendmail.cf). The procedure described here will therefore be done only with this one file and of course with the main NOD32LMS configuration file (/etc/nod32/nod32.cfg) .

First one has to grab information on the MDA used by Sendmail. This can be found in the Sendmail configuration file (/etc/mail/sendmail.cf) in the section called 'Local and Program Mailer specification' at the parameter 'P' (in the sentence starting with word 'Mlocal'). Yet, before performing any changes to the Sendmail configuration file, please, make a back-up of this file (for example with the name sendmail.cf.orig). Now let's return to the description of the sendmail.cf parameters. The parameter 'P' represents the full path to the MDA used. In order to involve 'nod32mda' module into the message processing, one has to replace this path by the appropriate 'nod32mda' module path (i.e. /usr/bin/nod32mda) . At the same time the appropriate change has to be done for parameter 'A' (located at the same sentence of Sendmail configuration file). Note that the 'A' parameter contains the file name of the original MDA component followed by some switches. These switches represent macros used by Sendmail when processing a message. Therefore none of the switches may be changed when modifying the name of the original MDA component at the parameter 'A'. For instance, if the original MDA component path is /usr/bin/procmail, the appropriate sentence in the Sendmail configuration file can look like this.

```
Mlocal,     P=/usr/bin/procmail, F=lsDFMAw5:/|@qSPhnu9,
            S=EnvFromL/HdrFromL, R=EnvToL/HdrToL,
            T=DNS/RFC822/X-Unix,
            A=procmail -t -Y -a $h -d $u
```

After the modifications have been performed, the appropriate sentence will look like this.

```
Mlocal,     P=/usr/bin/nod32mda, F=lsDFMAw5:/|@qSPhnu9,
            S=EnvFromL/HdrFromL, R=EnvToL/HdrToL,
            T=DNS/RFC822/X-Unix,
            A=nod32mda -t -Y -a $h -d $u
```

Note: Please, in case you are reading the ASCII form of this guide, do not drag and drop the above sentence, since it may not work. Indeed, it can be that the switches used here as an example will not work in your case as they are dependent on the version of Sendmail MTA.

With the above modifications we have ensured that all the messages originally sent by the MTA for delivery to the MAILBOX (by using an original MDA) will now be primarily sent to module nod32mda. Yet remains the second part of the modification to provide that all messages processed by nod32mda will be sent to the original MDA. In order to do so, we have to modify the parameter 'mda_path' within the section [mda] of the main NOD32LMS configuration file (/etc/nod32/nod32.cfg) to the original MDA component. The original MDA component can be, at this stage of the procedure, still found in the back-up file (/etc/mail/sendmail.cf.orig) you have created. Thus assuming your original MDA component is (/usr/bin/procmail) the appropriate parameter of the main NOD32LMS configuration file will look like this:

```
mda_path = "/usr/bin/procmail"
```

Note (for Debian users): The Debian distribution uses the following MDA: /usr/lib/sm.bin/sensible-mda. Note that this is not a full-blown MDA, it is rather a wrapper. However, the mda_path, in this case has the following format:

```
mda_path = "/usr/lib/sm.bin/sensible-mda"
```

To accomplish the whole procedure, one has to restart both the MTA Sendmail and the daemon nod32d.

5.2.2.2. Setting Postfix MTA

An MTA Postfix stores its configuration in several files. However, the modifications necessary to be described in this section will be done only with the main Postfix configuration file (/etc/postfix/main.cf) and of course with the main NOD32LMS configuration file (/etc/nod32/nod32.cfg) .

First one has to grab information on the MDA used by Postfix. This can be found by exploring the main Postfix configuration file (/etc/postfix/main.cf). To make all things easier, the authors of Postfix have written a helper utility to work with this file, so called 'postconf', so all the necessary steps will be done by using this utility. Also, before any further steps, please make a back-up of the original main Postfix configuration file (for instance, with the name /etc/postfix/main.cf.orig).

Now let's assume that the original MDA component is '/usr/bin/maildrop'. Then by using a command

```
postconf mailbox_command
```

we will receive the following information on return:

```
mailbox_command = maildrop -d "$USER" "$EXTENSION"
```

In the next step we will use the 'which' command

```
which maildrop
```

that will give us the full path to the binary file by return. Now, in order to involve module nod32mda into the message processing, we have to replace the MDA maildrop with the module nod32mda. Write the following command in order to do so:

```
postconf -e "mailbox_command = nod32mda -d \"$USER\" \"$EXTENSION\""
```

Note: Please keep in mind that the switches used in 'mailbox_command' after the MDA component must stay the same as in the original configuration file, otherwise the current configuration may not work properly.

With the above modifications we have ensured that all the messages originally sent by MTA for delivery to MAILBOX (by using an original MDA) will now be primarily sent to module nod32mda. Still there is the second part of modification to provide that all messages processed by nod32mda will be sent to the original MDA. In order to do so, we have to modify the parameter 'mda_path' within the section [mda] of the main NOD32LMS configuration file (/etc/nod32/nod32.cfg) to the original MDA component. The original MDA component can be at this stage of the procedure still found in the back-up file (/etc/postfix/main.cf.orig) you have created. Thus assuming your original MDA component is (/usr/bin/maildrop) the appropriate parameter of the main NOD32LMS configuration file will look like this:

```
mda_path = "/usr/bin/maildrop"
```

To accomplish the whole procedure, one has to restart both the MTA Postfix and the daemon nod32d.

5.2.2.3. Setting Qmail MTA

The delivery options in the Qmail program are configured via the command line parameters at the time of program execution or by using short scripts where the appropriate command line statements are stored. In the following we assume that the Qmail is installed in a /var/qmail directory and there is a Qmail's starting script called usually 'rc' within this directory that is always executed at the start-up of Qmail by running the standard statement

```
qmailctl start
```

In a simplest case the script (/var/qmail/rc) is of the following content.

```
#!/bin/sh

exec env - PATH="/var/qmail/bin:$PATH" \
qmail-start ./Maildir/ splogger qmail
```

Please note that this script is configured to deliver messages to a special directory structure ’./Maildir’ located within a local user home directory, as the first parameter of the qmail-start execution command is ’./Maildir/’. There are, indeed, more possibilities to be written at the place of the first command line parameter (for instance ./Mailbox to deliver the locally delivered messages to the MAILBOX file located at the local user home directory; ’|preline procmail’ to use the procmail MDA as a local deliver agent, etc.).

In order to involve the module nod32mda into the message delivery process the user is required to use: "|/usr/bin/nod32mda" as the first command line parameter, i.e. the starting script (var/qmail/rc) will look as follows:

```
#!/bin/sh

exec env - PATH="/var/qmail/bin:$PATH" \
qmail-start "|/usr/bin/nod32mda" splogger qmail
```

With the above modifications, we have provided that all the messages originally sent by the MTA for delivery to the MAILBOX (by using an original MDA) will now be primarily sent to module nod32mda. Still there is the second part of the modification to provide that all messages processed by nod32mda will be sent to the fully featured MDA agent that will take care about the message successful delivery to the MAILBOX. Here a problem may occur, as the original local delivery agent of Qmail is so called qmail-local. The qmail-local understands the structure of the Qmail starting script and is able to deliver e-mail messages to the special structures of MAILBOX, e.g. ’./Maildir/’ or ’./Mailbox’. However, this is not the case of other ’old featured’ MDAs, including procmail MDA. Moreover, the qmail-local delivery agent and the whole of Qmail uses different return codes of MDA than other standard MDAs and this has to be taken into account when configuring nod32mda module. In the following example we provide the modification that will use procmail MDA delivery agent, which will deliver the message to the MAILBOX as a special ’./Maildir/’ structure located at the appropriate user home directory. In order to do so, we will modify the parameters (infected_return_code, tmpfail_return_code) located in the section [mda] in the main NOD32LMS configuration file in the following manner:

```
infected_return_code = 100
tmpfail_return_code = 111
```

At the same time we will modify the parameter (mda_path) located at the same section [mda] of the same configuration file to use a custom script

```
mda_path = "/var/qmail/bin/qmail-maildrop"
```

that we will, of course, have to provide now. The /var/qmail/bin/qmail-maildrop, we have to create, will be as follows:

```
#!/bin/sh
```

```
/var/qmail/bin/preline /usr/bin/procmail && exit 0
[ $? = 75 ] && exit 111
exit 100
```

Note: Please check whether the path at each binary file used in the script corresponds to the real location of the binary file.

With the above modifications we have configured Qmail to send e-mail messages to nod32mda module from where they are after scanning sent to MDA procmail for local delivery. As has been indicated above, the procmail MDA, by default, delivers the messages to the /var/spool/mail/username directory unless it has been customized. The customization of procmail MDA can be done via a so called 'procmailrc' file. Procmail, by default, reads the configuration procmail file (/etc/procmailrc) and (.procmailrc located in user home directory) to get information where the message has to be delivered. Please read the appropriate manual pages of procmail and procmailrc to get detailed information on how to configure delivery options of procmail. In the next screen shot there is a short example of the procmailrc file used to deliver messages to the special ./Maildir/ directory structure.

```
SHELL=/bin/sh
LOGFILE=$HOME/proclog
VERBOSE=yes
PATH=/usr/local/bin
MAILDIR=$HOME/Maildir/
DEFAULT=$HOME/Maildir/
```

The above configuration file will tell procmail to deliver messages to the directory '$HOME/Maildir' where $HOME corresponds to the local user home directory and the procmail will also create a special output file '$HOME/proclog' where all its activity will be logged.

5.2.2.4. Setting MTA Exim version 3

An MTA Exim version 3 stores its configuration in the file /etc/exim/exim.conf (resp. /etc/exim.conf). The procedure described here will therefore be done only with this one file and of course with the main NOD32LMS configuration file (/etc/nod32/nod32.cfg) .

First one has to grab information on MDA used by Exim. This can be found in the main Exim configuration file (/etc/exim/exim.conf) in the section usually called 'TRANSPORTS CONFIGURATION'. However, first before any further steps, please make a back-up of the original main Exim configuration file (for instance with the name /etc/exim/exim.conf.orig). Now let's look inside the content of the appropriate section responsible for local delivery. This can, for instance, look like the following, where we assume the procmail be the original MDA component:

```
# TRANSPORTS CONFIGURATION
procmail_pipe:
  driver = pipe
  command = /usr/bin/procmail -d $local_part
  return_path_add
```

```
  delivery_date_add
  envelope_to_add
  check_string = "From "
  escape_string = ">From "
  user = $local_part
  group = mail

# DIRECTORS CONFIGURATION
procmail:
  driver = localuser
  transport = procmail_pipe
```

As can be seen in this screenshot, the original MDA component is referenced in the configuration file by parameter 'command'.

Note: Please note that besides the section related to transport there should always be defined the section usually called 'DIRECTORS CONFIGURATIONS' defined in the configuration file where the appropriate parameter transport is referenced.

In the next step we will use the 'which' command

```
which procmail
```

to get the full path of this MDA component. Now, in order to involve the module nod32mda into the message processing, we have to replace the MDA procmail by the module nod32mda. In order to do so, we are required to replace the appropriate line in the main Exim configuration file so it will look as follows:

```
command = /usr/bin/nod32mda -d $local_part
```

Note: Please keep in mind that the switches used in 'command' after the MDA component has stayed the same as in the original configuration file, otherwise the current configuration may not work properly.

With the above modifications we have ensured that all the messages originally sent by the MTA for delivery to the MAILBOX (by using an original MDA) will now be primarily sent to module nod32mda. Still there remains the second part of the modification to provide that all messages processed by nod32mda will be sent to the original MDA. In order to do so, we have to modify the parameter 'mda_path' within the section [mda] of the main NOD32LMS configuration file (/etc/nod32/nod32.cfg) to the original MDA component. The original MDA component can be, at this stage of the procedure, still found in the back-up file (/etc/exim/exim.conf) you have created. Thus assuming your original MDA component is (/usr/bin/procmail) the appropriate parameter of the main NOD32LMS configuration file will look like this:

```
mda_path = "/usr/bin/procmail"
```

To accomplish the whole procedure, one has to restart both the MTA Exim and the daemon nod32d.

5.2.2.5. Setting MTA Exim version 4

An MTA Exim version 4 stores its configuration in the file /etc/exim4/exim4.conf. The procedure described here will therefore be done only with this one file and of course with the main NOD32LMS configuration file (/etc/nod32/nod32.cfg) .

Please, note that the configuration described here is MDA independent by means it can be used in cooperation with any MDA already working with Postfix MTA. First before any further steps, please make a back-up of the original main Exim configuration file (for instance with the name /etc/exim4/exim4.conf.orig). Now let's look inside the content of the exim configuration file. It is typicaly compound from the TRANSPORTERS and ROUTERS sections. Usually there should be always ROUTER section responsible for messages local delivery (usually it is called 'loacaluser'). In order to configure exim to use NOD32LMS one has to define special ROUTER section

```
# ROUTER CONFIGURATION
nod32:
  driver = accept
  domains = +local_domains
  condition = "${if and { {!eq {$received_protocol}{virus-scanned}} } {1}{0}}"
  transport = nod32
```

and place this section before ROUTER section responsible for the local delivery.

One has also define appropriate TRANSPORTER section responsible to deliver e-mail messages to nod32mda agent.

```
# TRANSPORTS CONFIGURATION
nod32:
  driver = pipe
  command = /usr/bin/nod32mda $local_part@$domain
  return_path_add
  delivery_date_add
  envelope_to_add
  check_string = "From "
  escape_string = ">From "
  user = mailnull
  group = mail
```

With the above setting we have ensured that all the e-mail messages sent to user belonging to local domain will now be primarily sent to module nod32mda. Still there remains the second part of the modification to provide that all messages processed by nod32mda will be sent to the appropriate MAILBOX. In order to do so, we have to modify the parameter 'mda_path' within the section [mda] of the main NOD32LMS configuration file (/etc/nod32/nod32.cfg) in a following way.

```
mda_path = "/usr/local/sbin/exim-deliver"
```

Now it is necessary to create the script file '/usr/local/sbin/exim-deliver' with the following content.

```
#!/bin/sh
/usr/sbin/exim -oMr virus-scanned $*
```

In the above script we have asumed that the exim binary file is located within a directory /usr/sbin. If this is not the case, please modify above script as necessary. To acomplish whole procedure it is yet necessary to provide directory '/var/tmp/nod32' where exim will store temporary files concerned with newly defined configuration. Note that the directory has to be writable for the user 'mailnull', i.e. the following statements have to be written into command line.

```
mkdir /var/tmp/nod32
chown mailnull:mailnull /var/tmp/nod32
chmod 600 /var/tmp/nod32
```

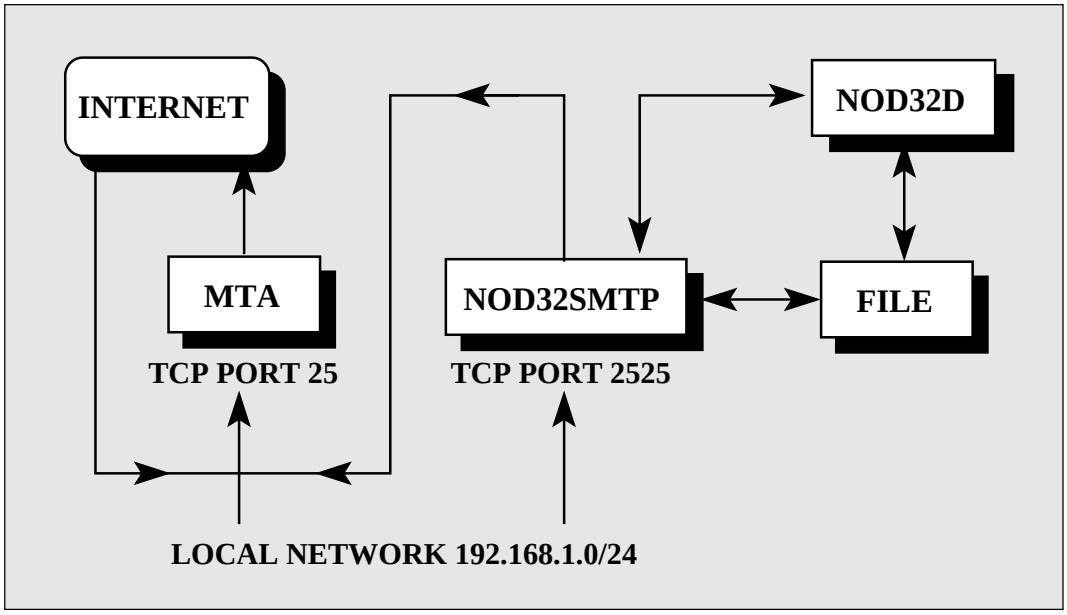
To accomplish the whole procedure, one has to restart both the MTA Exim and the daemon nod32d.

5.3. Scanning the outbound e-mail messages

Scanning of the outbound e-mail messages is performed at the time of message transmission between the local MUA and the MTA (as marked by symbol S2 in figure 3-1). A more detailed scheme of the process is shown in the figure 5-2.

Figure 5-2. The scheme of the scanning of outbound e-mail messages by using nod32smtp

module.



The most important part of scanning the outbound messages is done by the nod32smtp filter. This filter is a resident program (daemon) that performs in general three functions:

- receives data via the INET socket,
- extracts e-mail/s and feeds nod32d (scanning daemon) to scan it,
- forwards the e-mail to another port or computer.

The nod32smtp uses the SMTP protocol for its communication. As discussed in the section 5.1, the configuration of nod32smtp daemon is specified in section [smtp] of the main NOD32LMS configuration file /etc/nod32/nod32.cfg. The following parameters are defined:

listen_addr

specify the smtp server address where the daemon listens to

listen_port

specify the smtp server port where the daemon listens to

server_addr

specify the smtp server address where the communication will be redirected to

server_port

specify the smtp server port where the communication will be redirected to

The operation principle of scanning an outbound e-mail message is based on the following idea: We configure a nod32smtp daemon to listen to communication incoming to port 2525 of the e-mail server computer and forward the scanned communication to port 25 of the same computer where, typically, the MTA daemon listens to. In order to do so, use the following values of the aforementioned parameters:

```
listen_addr = "localhost"
listen_port = 2525
server_addr = "localhost"
server_port = 25
```

Once again, an e-mail received via port 2525 will be processed by nod32d (scanning daemon) and afterwards sent to port 25 for further processing by agent MTA of the e-mail messaging system. So far this is just half of the job. The second part that has to be done is the automatic redirection of all the packets arriving on port 25 of the server computer to port 2525. If we do not this then no packet of e-mail communication will go through the nod32smtp daemon as the whole communication from the LAN (Local Area Network) will pass by through port 25.

Thus, in order to do proper rerouting one has to write the following rules for ipchains (resp. iptables):

Kernel 2.2.*X*:

```
ipchains -I INPUT -p tcp -s 192.168.1.0/24 -d 0.0.0.0/0 25 \
      -j REDIRECT 2525
```

Kernel2.4.*X*:

```
iptables -I PREROUTING -t nat -p tcp -s 192.168.1.0/24 --dport 25 \
      -j REDIRECT --to-ports 2525
```

where ipchains (iptables) for kernel 2.2.X (2.4.X) have been used. After applying these rules and starting the nod32smtp daemon

```
/sbin/service nod32smtp start
```

one can see that all the communication arrives to the nod32smtp daemon and everything works correctly, however, with this setting there is still one unwanted effect. One can easily realize that the port 2525 with this setting provides an open relay. The point is that now there is a nod32smtp daemon that will accept all the packets that arrive at the port 2525, this also means packets arriving from outside the local network. The daemon nod32smtp will forward this traffic to port 25. This process will be interpreted by MTA as a local communication on the so called loop-back interface and therefore will not be rejected by MTA rules.

This problem, however, can be solved by ensuring that all communication with the port 2525 will be disabled with the exception of the local network. In order to do so we use the following commands:

Kernel 2.2.*X*:

```
ipchains -I INPUT -p tcp -s ! 192.168.1.0/24 -d 0.0.0.0/0 2525 -j REJECT
```

Kernel 2.4.X:

```
iptables -I INPUT -p tcp -s ! 192.168.1.0/24 --dport 2525 -j DROP
```

5.4. Content filtering in MTA

Content filtering method is currently a well known method used to screen and/or exclude certain defined information from the Internet or its part. Concerning an e-mail server system the best place to implement content filtering method is the MTA agent as an e-mail communication traffic nod. The advantage of such an implementation is that it allows one to scan e-mails inbound as well as outbound in the same implementation algorithm. On the other hand the content filtering method is MTA dependent. Taking into account the number of different MTAs working in the present UNIX OS environment, the method is not universal. NOD32LMS comes with two content filters built for most common MTA, i.e. MTA Sendmail and MTA Postfix, both described in the following sections.

5.4.1. Content filtering in MTA Postfix

The nod32smtp filter can also serve as a content filter for MTA Postfix providing the following changes are implemented:

In section [smtp] of the configuration file: /etc/nod32/nod32.cfg set:

```
listen_port = 2526
server_addr = "localhost"
server_port = 2525
```

With these settings nod32smtp will listen on port 2526 and will forward all communication from this port to the local port 2525.

In the next step, modify the /etc/postfix/master.cf file by adding the following specification into the file:

```
localhost:2525 inet  n - n - - smtpd
  -o content_filter=
  -o myhostname=nod32.yourdomain.com
```

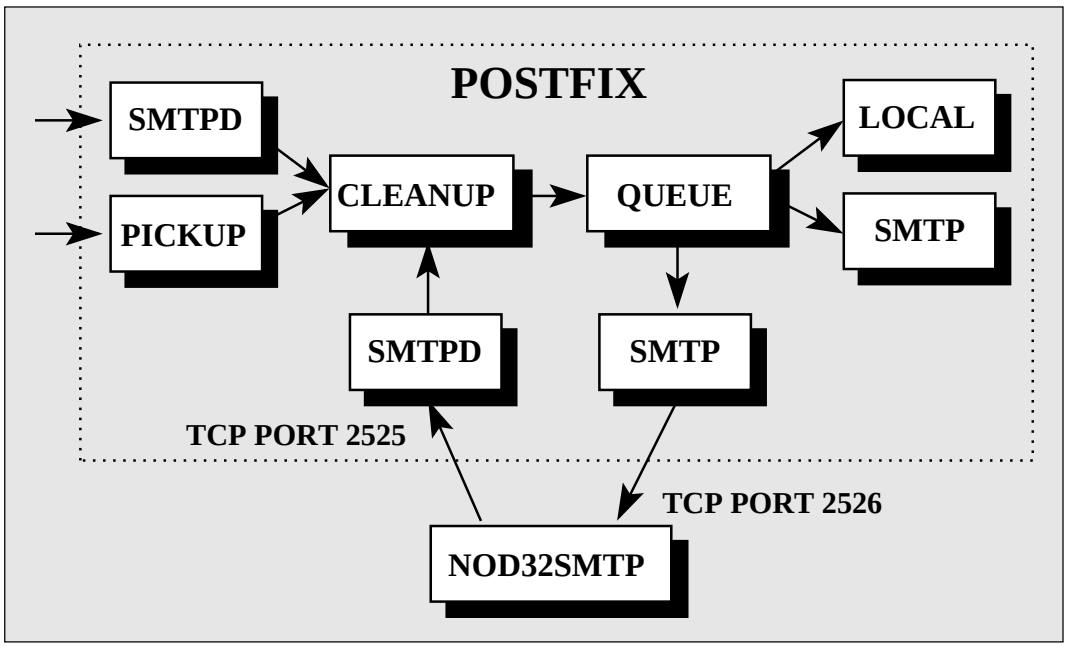
The value of the parameter myhostname must differ from that set in postfix (postconf myhostname), otherwise a loop is detected (by postfix) and the e-mail is rejected.

Finally, add

```
postconf -e "content_filter = smtp:localhost:2526"
```

into the postfix configuration file (/etc/postfix/main.cf). The entire process is illustrated in figure 5-3.

Figure 5-3. Bidirectional scanning scheme of an nod32smtp module working as a content filter.



5.4.2. Content filtering in MTA Sendmail

The nod32smfi serves as a content filter for MTA Sendmail. It is a third-party program that accesses all the mail messages being processed by MTA Sendmail and therefore can be used in the bidirectional scanning scenario.

In order to enable filtering, please provide the following line in the section [smfi] of the configuration file: /etc/nod32/nod32.cfg set:

```
smfi_sock_path = "/var/run/nod32smfi.sock"
```

With these settings nod32smfi will communicate with the MTA Sendmail via unix socket /var/run/nod32smfi.sock.

In the next step, modify the /etc/mail/sendmail.cf file by adding the following specification into the section MAIL FILTER DEFINITIONS:

```
Xnod32smfi, S=local:/var/run/nod32smfi.sock, F=T, T=S:2m;R:2m;E:5m
```

With this setting sendmail will communicate with the nod32smfi daemon via local (i.e. unix) socket /var/run/nod32smfi.sock. Flag F=T will result in temporary fail connection if the filter is unavailable. Flag T=S:2m defines timeout 2 minutes for sending information from MTA to a filter. Flag T=R:2m defines timeout 2 minutes for reading reply from the filter. Flag T=E:5m means overall timeout 5 minutes between sending end-of-message to filter and waiting for the final acknowledgment.

Note: In case the timeouts for the nod32smfi filter are set too small, Sendmail can temporarily reject the message which will attempt to pass through at a later time. This will lead to the continuous rejection of one and the same message later. In order to avoid the problem, the timeouts have to be set properly. In order to do this, one has to get into account 'confMAX_MESSAGE_SIZE' parameter defined in a sendmail.mc file that will provide not accepting messages bigger than the appropriate parameter value (given in bytes). Taking into account this value and the maximum time for processing of this amount of data by MTA (this can be measured) one can evaluate the appropriate timeouts for nod32smfi filter.

Finally, uncomment and modify the following line in the /etc/mail/sendmail.cf file:

```
O InputMailFilters=nod32smfi
```

Since nod32smfi filter can modify the content of the e-mail message body, in case of multiple Sendmail filters, it is good to put the definition of the nod32smfi filter at the end of the filter chain.

5.5. Alternative methods of scanning e-mails.

Although mechanisms described in previous sections are concerned to be the basic mechanisms of the e-mail messages scanning, there exists yet other possibilities that are all described in this section.

5.5.1. Scanning e-mail messages using AMaViS

AMaViS - A Mail Virus Scanner is a tool that interfaces your MTA and several antivirus scanners. It supports Sendmail, qmail, Postfix, Exim and comes in three branches:

amavis

for low/medium mail volume

amavisd

for higher mail volume, daemonized version of amavis

amavisd-new

for higher mail volume, Anti-Spam, ISP features, ...

Amavis works with NOD32LMS package using the command line interface nod32cli (see the nod32cli manual page for details). In this place it is good to discuss the differences of the NOD32LMS and Amavis configuration and performance as compared to the methods descibed in the previous sections. First it has to be born in mind that the Amavis does not allow to work with the original processed e-mail message. This means that all the manipulations with the e-mail copy by NOD32LMS will not affect the message delivered to the final recipient. This has of course some restrictive consequences to the NOD32LMS functionality. Particularily NOD32LMS is not able clean or delete the infiltrations from the e-mail message. Therefore configurations in the next subsections are done in a way that either Amavis will accept the message if it is originaly clean or it will drop the message in case it is non-clean. Further difference is in writing short strings into the message header or writing short text into the message footnote. All this functionality is invalid when using Amavis.

5.5.1.1. amavis

Configuration of Amavis with NOD32LMS is performed during the process of Amavis installation. For installation, first unpack the source amavis-0.x.y.tgz and overwrite the file amavis/av/nod32cli with this contents:

```
#
# ESET Software NOD32 Command Line Interface, Version 2.00
#

if ($nod32cli) {
    do_log(2,"Using $nod32cli");
    chop($output = ‘$nod32cli --subdir $TEMPDIR/parts‘);
    $errval = retcode($?);
    do_log(2,$output);
    if ($errval == 0) {         # no errors, no viruses found
        $scanner_errors = 0;
    } elsif ($errval == 1 || $errval == 2 || $errval == 10) {
                                # no errors, viruses discovered
        $scanner_errors = 0;
        @virusname = (undef);
        do_virus();
    } else {
        do_log(0,"Virus scanner failure: $nod32cli (error code: $errval)");
    }
}

# List of Return Codes
#define NOD32_EXIT_CODE_OK               0
#define NOD32_EXIT_CODE_CLEANED          1
#define NOD32_EXIT_CODE_DELETED          2
#define NOD32_EXIT_CODE_INFECTED         10
#define NOD32_EXIT_CODE_INTERNAL_ERROR   100
#define NOD32_EXIT_CODE_ERROR_IN_ARCHIVE 101
#define NOD32_EXIT_CODE_OPEN_ERROR       102
#define NOD32_EXIT_CODE_OTHER_STATUS     50
```

Next update your PATH issuing the command

```
export PATH="$PATH:/usr/bin"
```

For successfull installation you may need arc, unarj, unrar, zoo, make a symlink in /usr/bin from uncompress to gzip and create the user amavis in group amavis with home dir /var/amavis. Now continue with the usual installation proces (./configure, make, make install) and follow the rules README.mta according your mail server.

5.5.1.2. amavisd

Configuration of Amavisd with NOD32LMS is performed during the process of Amavisd installation. Unpack the source amavisd-0.x.tgz and follow the rules for amavis described in previous section of this guide.

Note: After 'make install' you may need to move /usr/etc/amavisd.conf to /etc and do a 'make install' again. Don't forget to run amavisd as user amavis after finishing the installation.

5.5.1.3. amavisd-new

In order to instal NOD32LMS with Amavisd-new, unpack and install the source amavisd-new-2.x.y.tgz in your installation directory. Now to configure NOD32LMS with newly installed Amavisd-new replace the clause for 'ESET Software NOD32 - Client/Server Version' with this one:

```
### http://www.nod32.com/
['ESET Software NOD32 Command Line Interface v 2.00',
  '/usr/bin/nod32cli', '--subdir {}',
  [0], [1,2,10], qr/virus names not available/ ],
```

You may need to install additional perl modules Archive-Tar, Archive-Zip, BerkeleyDB, Compress-Zlib, Convert-TNEF, Convert-UUlib, IO-stringy, MailTools, MIME-Base64, MIME-tools, Net-Server and Unix-Syslog from www.cpan.org/modules. The procedure is by each as follows: perl Makefile.PL; make; make install.

After configuring NOD32LMS follow the recommendation for configuring Amavisd-new in README.mta located in Amavisd-new directory according your mail server.

Chapter 6. NOD32 system update and maintenance

In order to keep NOD32LMS system effective, it is necessary to keep NOD32 virus definitions database up to date. In the following sections a concept of the NOD32 database update process is described with more in depth details which is in most cases not necessary to read. Therefore one may wish to skip directly to section 6.3 that can serve in this sense as a short 'how-to' document related to this topic.

6.1. Basic concept of NOD32 system update.

Basic concept of the NOD32 system update is to download so called NOD32 modules from the NOD32 server and compile them in order to provide so called loading modules used by the NOD32 anti-virus scanner. First part of the process is provided by NOD32 Update Mirror Creator (nod32umc) system utility while compilation of the modules is done by NOD32 Update (nod32upd) system utility. Both the utilities are part of the NOD32LMS system.

6.1.1. NOD32 modules organization

The NOD32 modules introduced above are located by default within

```
/var/lib/nod32/mirror
```

directory (so called local mirror directory or default local mirror directory). These modules are divided into two categories, e.g. engine category and component category (modules of component category are currently only for use on the Windows operating systems).

Currently there are 4 types of engine category modules supported:

- base scanning modules (prefix engine) containing virus signatures database,
- archives support modules (prefix archs) supporting various file system archive formats,
- advanced heuristics modules (prefix advheur) containing implementation of so called advanced heuristics method of virus and worm detection,
- packed worm scanner modules (prefix pwscan).

These modules are always necessary for proper running of any NOD32 anti-virus scanner based application and therefore are all downloaded by default at each download process.

On the other hand the component category modules are platform dependent and language localization dependent and thus the selection of them has to be specified by the user according to his needs. Download of component category modules is therefore optional.

Besides the NOD32 modules, the default mirror directory contains so called modules version control file (update.ver). This file contains relevant information concerning all modules stored

in the local mirror directory and is used as a reference file for generation of further subordinate mirrors. The process of subordinate mirror creation is described in the section 6.2 in detail.

6.1.2. NOD32 mirror creation

As has already been told, NOD32 update mirror creator (nod32umc) utility is used to download and maintain the mirror of NOD32 modules storage. Minimal nod32umc configuration requires that you define authentication options and the absolute path to the directory where the mirror will be created.

Authentication options are used to pass to nod32umc the username and password (received from vendor) necessary to authenticate the end-user against NOD32 server where the modules are downloaded from. Several scenarios for user authentication against NOD32 server are supported:

- direct username and password setting into the command line.
- setting via an authentication file,
- setting via an old featured '.netrc' file.

Please refer also to the nod32umc manual pages for a detailed description of the related command line options.

After proper authentication the modules will be downloaded into a defined mirror directory.

There exists also the possibility to download and update NOD32 modules in the situation where all the communication of the local computer with the NOD32 server is mediated via http proxy server. Read more on this topic in nod32umc manual pages. In situations when user authentication is required against the appropriate proxy server, the minimal nod32umc configuration must include also the proxy authentication options. There are two supported scenarios for user authentication against proxy server:

- direct proxy username and proxy password setting into the command line
- setting via a proxy authentication file.

Please, refer also to the nod32umc manual pages for a detailed description of the related command line options.

Note: For security reasons, please use authentication scenarios via authentication files rather than scenarios via typing the username(s) and password(s) directly into the command line.

6.1.3. Generation of NOD32 scanner loading modules

In order to create loading modules from those downloaded by nod32umc one must use the NOD32 Update (nod32upd) system utility (see also nod32upd manual pages). Minimal configuration for nod32upd is to define full mirror directory path. The following NOD32 loading modules will be created: base module (nod32.000), archives support module (nod32.002), advanced heuristics module (nod32.003) and packed worm scanner module (nod32.004). One could specify the directory path where the loading module would be created (so called NOD32

base directory, i.e. directory where the NOD32 anti-virus scanner will load the modules from). One can choose an arbitrary directory for this purpose, however, this must contain nod32.000 module. There is no requirement given on the version of the module. In case the module is not located in the directory chosen, error 105 will occur. It is recommended to set NOD32 base directory as '/var/lib/nod32' as this is exactly the default directory where the modules are loaded by NOD32LMS anti-virus scanner from. In case the NOD32 base directory is not chosen by the user it is defined by default as the current working directory.

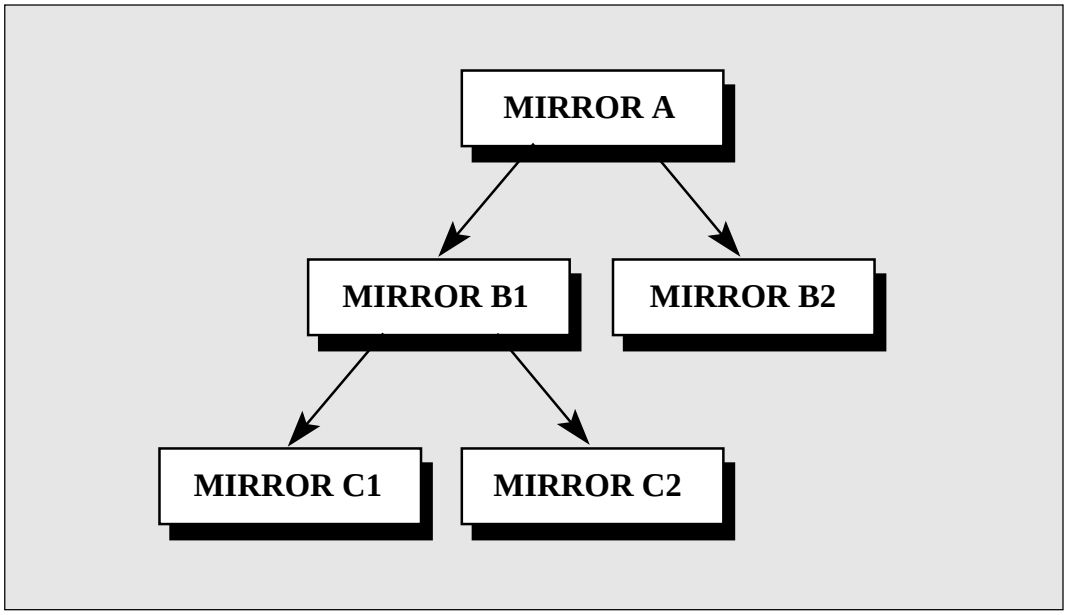
In case the 'update.ver' file located in the mirror directory is not signed by valid digital signature, error 107 will appear. One can force nod32upd to not check the digital signature of the file by choosing switch '--no_signature'.

Please refer to the manual pages nod32upd to get detailed information on the command line options.

6.2. Subordinate mirrors creation

After download of NOD32 modules the nod32umc utility will create in the mirror directory the 'update.ver' file containing the information about the modules currently stored in the newly created mirror. The newly created mirror is a fully functional mirror of NOD32 modules storage and can be used in the process of additional mirrors creation. This hierarchy allows the user to update all the modules in all the subsequently created mirrors without using, and thus overloading, one and the same mirror server (see figure 6-1 for details).

Figure 6-1. The scheme of subordinate NOD32 mirrors creation. With the presented hierarchy it is not necessary to perform update process of the mirror C1 by tasking the mirror A which preserves the appropriate server from the overload.



At this point we would like you to notice that the creation of the mirror file structure at some computers file system, is not enough to provide the fully featured NOD32LMS mirror. For proper function of the newly created mirror there are additional conditions to be fulfilled. First, as the utility nod32umc uses http protocol to download the NOD32 modules there must be a http server installed on the computer where the modules are going to be downloaded from. Second, the NOD32 modules to be downloaded by other computers have to be placed at the directory path

```
/http-serv-base-path/nod_upd
```

where http-serv-base-path is a base http server directory path, as this is the first place where nod32umc utility looks the NOD32 modules for.

6.3. Automatic update of the virus definitions database

The NOD32LMS comes with the script

```
/usr/sbin/nod32_update
```

used to provide user with a comfortable and reliable service concerned with the process of NOD32 virus signatures database update.

6.3.1. Structure and use of automatic update script

The script is predefined to download NOD32 modules from the server

```
http://www.nod32.com
```

and to create the loading modules used by NOD32 scanner into

```
/var/lib/nod32
```

directory. This configuration can be, however, easily changed by simple editing the script in its configuration part.

Even if the script does all the job in most cases without manual intervention into it, one has still to define the username and password for NOD32 server authentication prior to using it. By default the update script reads this setting from the so called 'authorization file'

```
/etc/nod32/nod32.auth
```

that has to be adjusted by the user. The authorization file is an ASCII file with the following format.

```
username=your_nod32_username
password=your_nod32_password
```

Please, use your favorite editor to invoke the file and fill its appropriate parts by valid username and password that you have received from your vendor.

From the security reason make sure that the authorization file is publicly unreadable. If it is not the case change its permission appropriately by typing

```
chmod 600 /etc/nod32/nod32.auth
```

into the command line.

6.3.2. Periodic update of the virus definitions database

To provide the highest security for the user, the NOD32 team collects the virus definitions continuously from all over the world. The new patterns can appear within the database in very short intervals. It is therefore useful, and also recommended, to trigger an update attempt (by means of executing the nod32_update script) on a regular basis via periodic scheduler. One hour interval is recommended. In order to do so, configure periodic scheduler (cron) on your system by entering the following line

```
0 * * * * /usr/sbin/nod32_update
```

into its configuration file (crontab). To add the above line into the crontab use command line statement

```
crontab -e
```

that will invoke the editor set up for the current system environment (defined by EDITOR environment variable).

Chapter 7. Tips and tricks

This chapter is devoted to describe tips and tricks cocnerned with configuration of NOD32LMS. This means it describes configuration of NOD32LMS in circumstances when for instance MTA is configured to use other software with similar functionality or with functionality that could normally lead to misconfiguration of NOD32LMS.

7.1. Dropping messages marked by NOD32 as deleted in MTA Postfix

In an Internet has recently appeared nonnegligible increase of the number of the e-mail messages containing worms. In most cases the infected attachment of such messages cannot be cleaned but rather deleted and whole messages even does not contain any reasonable information. In such a cases it has sense to discard (or treat in special way) this kind of messages. Mechanism described in this section can be used to suppress messages marked by NOD32LMS as deleted in MTA Postfix.

First of all one has to add the following entry

```
write_to_header = 1
```

into section [smtp] of the main NOD32LMS configuration file (/etc/nod32/nod32.cfg) . This setting will result in a modification of each non-clean e-mail message by means the string 'X-NOD32Result: status' is inserted into header of the message. Word 'status' of the string is replaced by actual status of the scanning process.

In order to discard all messages that has been marked as 'deleted' one has to add following line

```
header_checks = regexp:/etc/postfix/header_checks
```

into the main Postfix configuration file (/etc/postfix/main.cf). At the same time one has to create file '/etc/postfix/header_checks' with the following content.

```
/^X-NOD32Result: deleted/ DISCARD
```

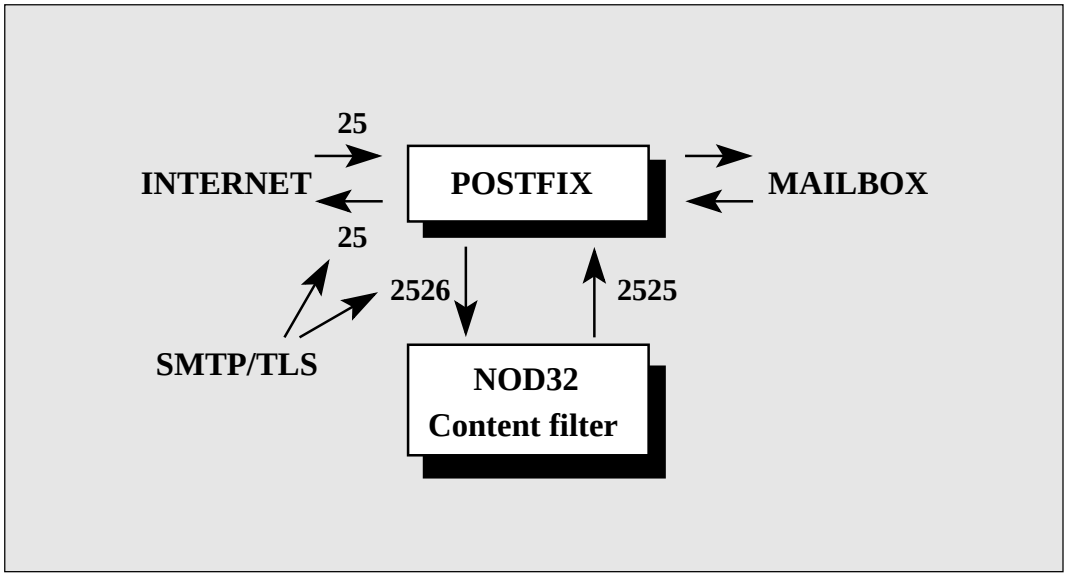
To accomplish the whole procedure, one has to restart MTA Postfix, daemon nod32d and daemon nod32smtp.

Note that in older Postfix versions DISCARD functionality may not work. In this case warning message 'Postfix does not know the command DISCARD' will appeare in the '/var/log/maillog' file. This can be only solved by update of the Postfix package.

7.2. NOD32LMS and TLS support in MTA

Transport Layer Security (TLS) is a protocol guaranting data privacy in client/server communication over the Internet. The basic principle of TLS is based on the SSL encryption of data traveling between client and server (We have on our mind the SMTP communication between MTA client and server). This has of course nonnegligible consequences for scanning of this kind of communication by NOD32LMS. For instance, once TLS support in MTA is enabled, the 'outbound messages scanning scenario' discussed in section 5.3 is impossible as the whole intercepted SMTP communication is encrypted at this stage. On the other hand, there is possibility to use data encryption in communication between local MTA and Internet and still use the NOD32LMS as a content filter (discussed in section 5.4). In MTA Sendmail content filtering there is no problem with SMTP TLS support at all as the Sendmail Milter does not relay on the SMTP communication and content filtering is done rather internaly. On the other hand the Postfix uses SMTP protocol for data communication between content filter and MTA. Therefore once the TLS is enabled in Postfix, the content filtering method fails as whole the SMTP communication is encrypted. Fortunatelly, this can be soloved on the Postfix TLS configuration level. The situation is depicted in a figure 7-1.

Figure 7-1. Scheme of content filtering in Postfix MTA with enabled TLS.



As is shown in the figure above, once the TLS is enabled, all the SMTP communication channels including SMTP communication with content filter are affected. The only possibility in this case is to disable the TLS support for communication between client and server located within localhost. This can be achieved by adding the following line into the main Postfix configuration file.

```
smtp_tls_per_site = hash:/etc/postfix/smtp_tls_per_site
```

In addition it is necessary to create the above file with the following content

```
localhost      NONE
```

and provide its appropriate hash table. In order to do so, execute the following statement from '/etc/postfix' directory.

```
postmap hash:smtp_tls_per_site
```

By using the above statement the '/etc/postfix/smtp_tls_per_site.db' file is created that is used by Postfix to enable TLS on per site basis. As far as we have disabled TLS for localhost the content filtering can be used and at the same time the SMTP communication between local MTA and Internet is encrypted.

Chapter 8. Let us know

Dear user, this guide should have given you a good knowledge about how the NOD32LMS works and how to configure it in order to protect your e-mail messaging system with highest efficiency. However, writing a documentation is a process that is never finished. There will always be some parts of the NOD32LMS that can be explained better or are not even explained at all. Therefore, in case of bugs or inconsistencies found within this documentation, please report a problem to our support center

`http://www.nod32.com/support/support.htm`

or write an e-mail directly to linux development mailing list

`<linux@nod32.com>`.

Appendix A. Installed content of NOD32LMS package

```
/etc/init.d/nod32d
/etc/init.d/nod32smfi
/etc/init.d/nod32smtp
/etc/nod32/nod32.auth
/etc/nod32/nod32.cfg
/etc/nod32/nod32d_script
/etc/nod32/sig_footer_clean.html.example
/etc/nod32/sig_footer_infected.html.example
/etc/nod32/sig_header_clean.html.example
/etc/nod32/sig_header_infected.html.example
/usr/bin/nod32mda
/usr/bin/nod32smfi
/usr/bin/nod32smtp
/usr/bin/nod32cli
/usr/sbin/nod32_update
/usr/sbin/nod32d
/usr/sbin/nod32umc
/usr/sbin/nod32upd
/usr/share/doc/nod32lms/copyright
/usr/share/doc/nod32lms/guide.us.txt.gz
/usr/share/man/man1/nod32mda.1.gz
/usr/share/man/man1/nod32smfi.1.gz
/usr/share/man/man1/nod32smtp.1.gz
/usr/share/man/man1/nod32cli.1.gz
/usr/share/man/man5/nod32.5.gz
/usr/share/man/man5/nod32.cfg.5.gz
/usr/share/man/man8/nod32d.8.gz
/usr/share/man/man8/nod32umc.8.gz
/usr/share/man/man8/nod32upd.8.gz
/var/lib/nod32/mirror
```